VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Webové služby a dynamické programovací jazyky
Web Services and Dynamic Programming Languages

2013

Tomáš Richter

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Tomáš Richter**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Webové služby a dynamické programovací jazyky
Web Services and Dynamic Programming Languages

Zásady pro vypracování:

Cílem práce je zmapovat ve vybraných dynamických jazycích (PHP, Python, Ruby) možnosti implementace a použití webových služeb se zaměřením na protokol SOAP.

1. Popište možnosti použití webových služeb ve vybraných dynamických jazycích (integrované funkce, popř. nejrozšířenější přídavné knihovny).
2. Naimplementujte server a klienta webové služby ve vybraných jazycích a v prostředí .NET (v některém jeho nedynamickém jazyce).
3. Experimentálně prověřte možnosti vzájemné spolupráce vybraných jazyků/knihoven. Výsledky dokumentujte a sestavte doporučení, podle kterých lze případné problémy spolupráce vyřešit. Kromě standardního posílání zpráv prověřte i posílání SOAP hlaviček.

Seznam doporučené odborné literatury:

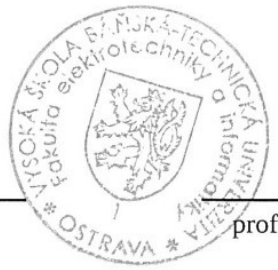
[1] Louridas, P. SOAP and Web Services. <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4012625>
[2] W3C SOAP Version 1.2. <http://www.w3.org/TR/2007/REC-soap12-part1-20070427/>
Vaughan-Nichols, S.J. Web services: beyond the hype. 2002
[3] <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=982908>
[4] PHP Manual, SOAP. <http://php.net/manual/en/book.soap.php>
[5] PythonInfo Wiki, WebServices. <http://wiki.python.org/moin/WebServices>
[6] The Ruby Toolbox, SOAP Clients. <https://www.ruby-toolbox.com/categories/soap>
[7] Curbera, Francisco, et al. "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI." Internet Computing, IEEE 6.2 (2002): 86-93

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jakub Macek**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: .....

podpis studenta

Poděkování

Rád bych poděkoval *Ing. Jakubu Mackovi* za odbornou pomoc a konzultaci při vytváření této diplomové práce.

Abstrakt

Bakalářská práce se zabývá problematikou implementace a používání webových služeb, se zaměřením na protokol SOAP, ve vybraných dynamických jazycích PHP, Python, Ruby a v prostředí .NET. První část práce obsahuje seznámení s webovými službami. Druhá část práce je věnována popisu WSDL dokumentu. Následující třetí část se zabývá SOAP protokolem. Čtvrtá část práce popisuje dostupné knihovny pro SOAP protokol ve vybraných dynamických jazycích. Poslední oddíl práce popisuje implementaci SOAP protokolu ve vybraných programovacích jazycích a prostředí .NET. V poslední části je uveden postup pro posílání SOAP zpráv a pokus o posílání SOAP hlavičky.

Klíčová slova

Webové služby, SOAP, WSDL, .NET, Microsoft Visual Studio, PHP, Python, Ruby, SOAP server, SOAP klient, SOAP zpráva, SOAP hlavička

Abstract

This thesis deals with problems of implementation and use of web services, with a focus on the SOAP protocol, in selected dynamic languages PHP, Python, Ruby and .NET environment. The first part contains web services introduction . The second part is devoted to describing the WSDL document. The third part is working with the SOAP protocol. The fourth part describes the available libraries for SOAP protocol in selected dynamic languages. The last part describes the implementation of the SOAP protocol in the selected programming language and NET environment. This section describes the procedure for sending SOAP messages and attempt to send SOAP headers.

Key words

Web services, SOAP, WSDL, .NET, Microsoft Visual Studio, PHP, Python, Ruby, SOAP server, SOAP client, SOAP message, SOAP header

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
SOAP	Simple Object Access Protocol	Jednoduchý objektový přístupový protokol
WSDL	Web Services Description Language	Jazyk popisující webové služby
RPC	Remote procedure call	Vzdálené volání procedur
XML	Extensible Markup Language	Rozšiřitelný značkovací jazyk
HTTP	Hypertext Transfer Protocol	Hypertextový protokol
W3C	World Wide Web Consortium	Mezinárodní konsorcium
SOA	Service-Oriented Architecture	Architektura orientovaná na služby
UDDI	Universal Description Discovery and Integration	Univerzální popis objevů a integrace
API	Application Programming Interface	Rozhraní aplikačního programovaní

Obsah

1 Úvod

Výsledky této bakalářské práce umožní ušetřit čas při hledání informací, jak v dynamických jazycích PHP, Python, Ruby a v prostředí .NET, protokol SOAP implementovat. Bakalářská práce seznamuje s webovými službami pomocí SOAP protokolu, který popisuje.

Práce shrnuje informace o různých knihovnách pro SOAP protokol a jejich možném využití. Ve vybraných SOAP knihovnách a dynamických jazycích pojednává o vzájemné kompatibilitě mezi těmito jazyky za použití SOAP protokolu. Implementuje server a klienta ve všech vybraných dynamických jazycích a prostředí .NET. Obsahuje posílání SOAP zpráv a prověření posílání SOAP hlaviček. Posílání SOAP zpráv vyzkouším několika způsoby. Posílání SOAP zpráv bez parametru , s parametrem a několika parametry. Tyto posílané parametry budou obsahovat číslo, nebo textovým řetězec.

Návody pro instalaci použitých knihoven jsou uvedeny v příloze. Úplnou implementaci (zdrojových kódů) jednotlivých příkladů možno nalézt v přiloženém souboru.

Pro práci v prostředí .NET jsem využil jazyk C# v programu Microsoft Visual Studio 2010.

Následně je možné podle sepsaných doporučení a postupů shrnutých v této bakalářské práci zprovoznit server a klienta webové služby ve vybraných jazycích a v prostředí .NET přes protokol SOAP.

2 Webové služby

Informace o této kapitole jsou převzaty převážně z webové stránky [1], *Wikipedie: otevřená encyklopedie,* http://cs.wikipedia.org/wiki/Webová_služba.

Webová služba je softwarový systém umožňující komunikace dvou strojů na síti (internetu). Pokud se jedná o webovou službu přes SOAP protokol, může být popsána pomocí WSDL dokumentu, který popisuje veškerou funkčnost webové služby. Z WDSL dokumentu můžu zjistit, jaké funkce s jakými parametry a jakou návratovou hodnotou lze přes tuto webovou službu volat.

Jako další možnost pro práci s webovými službami je možné použít jednodušší protokol XML-RPC, nebo složitější rozhraní REST, které je na rozdíl od známějších XML-RPC či SOAP, orientován datově, nikoli procedurálně. REST využívá pro komunikaci metod protokolu HTTP (POST, PUT, DELETE, ...). Existují i další přístupy k webovým službám, jako CORBA, GIOP nebo DCOM.

Webové služby umožňují snadnou spolupráci programů běžících na libovolných platformách.

Protokoly SOAP a WSDL jsou oba provedeny v syntaxi jazyka XML, proto můžou požadavky a odpovědi zpracovávat stejně jako jakékoliv XML.

Jiné systémy komunikují s SOAP webovou službou předepsaným způsobem pomocí SOAP zpráv, typicky zprostředkovaným pomocí protokolu HTTP s XML.

Webová služba je reprezentována softwarovým, nebo hardwarovým agentem, kteří mohou být různí, ale webová služba samotná zůstává stále nezměněná. Během transakce jeden agent žádá o službu a druhý službu poskytuje. Standardy používané specifikací webových služeb mají za úkol zajistit totožnou sémantiku obou agentů.

2.1.1 WSDL

Informace o této kapitole jsou převzaty z webových stránek [2],[3],[4].

Wikipedie: the free encyclopedia, 2.5.2013 http://en.wikipedia.org/wiki/WSDL

W3schools.com, © 1999-2013 http://www.w3schools.com/wsdl/

W3C, 15.3.2001 http://www.w3.org/TR/wsdl

WSDL je zkratka pro Web Services Description Language (jazyk popisující webové služby) je založený na XML a je v podstatě XML dokument. WSDL popisuje webovou službu, naleznu v něm informace, například funkce dané webové služby, spolu s formátem zprávy a detaily protokolu, jaké parametry funkce očekává a jaké struktury vrací. Říká, jak pracovat s danými webovými službami. Zpravidla popisuje SOAP komunikaci pro poskytování webových služeb po internetu.

WSDL popisuje služby jako kolekci síťových koncových bodů nebo portů. WSDL specifikace poskytuje XML formát za tímto účelem. V abstraktní definici portu a zpráv jsou rozděleny od jejich konkrétních použití nebo instancí umožňující opětovné použití těchto definic. Port je definován sdružením síťových adres s opakovanou vazbou a kolekcí portů určujících službu. Zprávy jsou abstraktně popsané údaje, které byly vyměněny a typy portů jsou abstraktními kolekcemi podporovaných operací. Konkrétní protokol a specifikace datového formátu pro daný typ portu vytváří opakovanou vazbu, kde jsou operace a zprávy umístěny v konkrétním síťovém protokolu a formátu zprávy. Tímto způsobem WSDL popisuje veřejné rozhraní pro webové služby.

2.1.2 Elementy WSDL

Dokument WSDL popisuje webovou službu pomocí těchto hlavních elementů:

`<definitions>` ve verzi WSDL 1.1, `<description>` ve verzi WSDL 2.0
- `<types>`
  Kontejner pro definici typu dat používaných webovou službou.
- `</types>`
- `<message>` ve verzi WSDL 1, ale ve verzi WSDL 2.0 je tento element odstraněn
  Definice dat pro komunikaci.
- `</message>`
- `<portType>` ve verzi WSDL 1.1, `<interface>` ve verzi WSDL 2.0
  Soubor podporovaných činností z jednoho či více koncových bodů.
- `</portType>`
- `<binding>`
  Specifikace protokolu a datového formátu pro konkrétní typ portu.
- `</binding>`

`</definitions>`

WSDL může také obsahovat další elementy, jako je „extension elements“ a servisní prvek, který umožňuje seskupit definice několika webových služeb v jediném dokumentu WSDL.

2.1.3 Verze WSDL

WSDL 1.0 byl vyvinut společností IBM, Microsoft, a Ariba kolem září 2000 k popsání webových služeb pro jejich SOAP nástroj. Byl vytvořen kombinací dvou jazyků pro popis služby: NASSL (Network Application Service Specifikace Language) od IBM a SDL (Service Description Language) od společnosti Microsoft.

WSDL 1.1 je formalizace předchozí verze WSDL, WSDL 1.1 byla publikovaná v březnu 2001. Žádné zásadní změny nebyly zavedeny. V tomto roce bylo poprvé zdokumentováno, jak používat WSDL ve spojení s SOAP 1.1 a HTTP GET / POST.

Význam zkratky WSDL se změnil od verze 1.1, kde D představovalo Definition (definujcí).

WSDL 1.2 a jeho první pracovní návrhy byly zveřejněny 17. prosince 2001. Poslední pracovní návrhy byly zveřejněny 11. června 2003. WSDL 1.2 je jednodušší a flexibilnější pro vývojáře než předchozí verze. Definuje lépe HTTP 1.1 vazbu než verze předešlá.

WSDL 2.0 se stal W3C doporučením formátem 26. června 2007. Vznikl přejmenováním WSDL verze z 1.2 na 2.0, protože má značné rozdíly oproti WSDL 1.1. Hlavním rozdílem jsou jiné názvy jednotlivých elementů. WSDL 2.0 byl navržen k řešení problému schopnosti vzájemné spolupráce různých systémů, ale tento záměr ještě není zcela dokončen, a proto se WSDL 2.0 ještě moc často nevyužívá. Jedná se o lepší volbu, pokud jde o vymezení RESTful služby.

Odstraněný element <message> a definován vevnitř elementu <operation>. Přetěžování operátorů není podporováno, <definitions> je přejmenován na <description>. <PortType> přejmenován na <interface> (rozhraní). <Port> přejmenován na <endpoint> (koncový bod).

2.2 SOAP

Informace o této kapitole jsou převzaty z webových stránek [5],[6],[7].

W3C, 27.4.2007 http://www.w3.org/TR/2007/REC-soap12-part1-20070427/

W3schools.com, © 1999-2013 http://www.w3schools.com/soap/

Wikipedie: the free encyclopedia, 18.4.2013 http://en.wikipedia.org/wiki/SOAP

SOAP původní označení **Simple Object Access Protocol** je protokol určený pro výměnu strukturovaných informací a pro implementaci webových služeb v internetové síti.

SOAP je nástupce protokolu XML-RPC, ačkoliv si zapůjčuje jeho způsob přenosu dat a další vlastnosti. Struktura komunikace obálka, hlavička a tělo (envelope/header/body) je převzata pravděpodobně z WDDX (Web Distributed Data eXchange).

Protokol SOAP tvoří základní vrstvu komunikace mezi webovými službami a poskytuje prostředí pro tvorbu složitější komunikace.

Existuje několik různých druhů šablon pro komunikaci pomocí SOAP protokolu. Nejznámější z nich je RPC šablona, kde jeden z účastníků komunikace je klient a na druhé straně je server. Server ihned odpovídá na požadavky klienta.

SOAP poskytuje způsob komunikace mezi aplikacemi běžícími na různých operačních systémech používajících různé technologie a programovací jazyky. SOAP protokol je rozšiřitelný v rámci webových služeb a může být použit v kterémkoli transportním protokolu, jako HTTP, SMTP, TCP nebo JSM.

2.2.1 Historie

Protokol navrhly Dave Winer, Don Box, Bob Atkinson a Mohsen Al-Ghosein v roce 1998 za podpory firmy Microsoft. Dnes je specifikace SOAP protokolu držena XML skupinou tvořící internetové protokoly z W3C konsorcia.

SOAP původně kandidoval na "Simple Object Access Protocol", ale od této zkratky bylo upuštěno s verzí 1.2 standardu, která se stala doporučením W3C od 24. června 2003. Zkratka SOAP je někdy zaměňována s architekturou SOA.

Po prvním uvedení protokolu se stala základní vrstva SOAP složitější sadou webových služeb založených na Web Services Description Language (WSDL) a Universal Popis Discovery a integrace (UDDI). O službu UDDI, se ukázal být velmi malý zájem.

2.3 Formy přenosu

Protokol SOAP může jako aplikační vrstvu použít HTTP i SMTP, ale protokol HTTP je daleko více používaný. Je tomu tak především proto, že HTTP je prakticky v internetu nejpoužívanější strukturou. Proto může SOAP jednoduše procházet přes firewall. Tento princip je hlavní výhodou oproti jiným podobným protokolům, které jsou většinou na firewallu zakázány (jako například DCOM).

XML formát byl zvolen jako standard pro přenos SOAP zpráv hlavně pro jeho rozšířenost a dostupnost vývojových nástrojů nabízených jako opensource nebo freeware-umožňuje tedy jeho volné používání.

Zdlouhavá syntaxe XML má své výhody i nevýhody. Výhodou je jednodochost, ale počítač ji musí složitě parsovat. Parsování zabírá hodně procesorového času a operační paměti. Oproti tomu CORBA, GIOP nebo DCOM má zápis zpráv pro komunikaci daleko kratší a binární. Na druhou stranu vývoj počítačů jde rychle dopředu a tak parsování přestává problémem.

2.4 Příklad použití

SOAP zpráva může být odeslána na webové stránky, které mají povolené webové služby, jako je například nějaká databáze (pozemků), s parametry potřebnými pro hledání. Stránka by pak vrátila dokument ve formátu XML s výslednými daty, např. ceny, umístění, funkce. Získané údaje se vrátí do standardizovaného formátu, který potom lze integrovat přímo do webové stránky nebo aplikace.

2.5 Výhody

SOAP je dostatečně univerzální, aby umožnila požití různých transportních protokolů, jako HTTP nebo JMS a SMTP.

Vzhledem k tomu SOAP model pracuje s HTTP, může snadno projít přes stávající firewally a proxy bez úprav na protokolu SOAP.

2.6 Nevýhody

Pomalejší zpracování, způsobené použití XML parsovaní a validace.

Velký zápis komunikace a větší složitost.

2.7 Ukázka

Příklad ukazuje velmi jednoduchý SOAP požadavek, který posílá jméno na server a ten vypíše pozdrav. Ukázka obsahuje pouze posílání SOAP zprávy. V elementu body se nachází požadavek na vyvolání vzdálené funkce "Ahoj" s parametrem pojmenovaným "kdo" s hodnotou "Lucie".

Požadavek:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://example.com/sample.wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <ns1:Ahoj>
      <kdo xsi:type="xsd:string">Lucie</kdo>
    </ns1:Ahoj>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Odpověď:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <ns1:AhojResponse xmlns:ns1="http://example.com/sample.wsdl">
      <return xsi:type="xsd:string">Ahoj Lucie.</return>
    </ns1:AhojResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2.7.1 Tvorba SOAP zprávy (SOAP message contstruct)

Definuje strukturu SOAP zprávy, která musí odpovídat XML serializaci. Jmenné prostory by měly být "namespace-qualified", jako v XML. **SOAP zpráva (SOAP message)** je základní jednotka mezi SOAP uzly. SOAP uzel ztělesňuje zpracování logiky potřebné pro přenos, příjem, zpracování, nebo předání SOAP zprávy.

2.7.1.1 SOAP obálka (SOAP envelope)

Obálka je element, který definuje strukturu zprávy a jak jí zpracovat. Naleznete v ní sadu kódovacích pravidel pro zpracování datových typů definovaných aplikací. Rovněž obsahuje konvenci pro volání procedur a také odpovědi.

Obsahuje:

- Místní název[local name] obálky
- Jmenný prostor[namespace name] "http://www.w3.org/2003/05/soap-envelope"
    - Jestli je použitý jiný, aplikace vygeneruje chybu a zprávu zahodí
    - EncodingStyle atribut se používá k definování datového typu použitého v dokumentu, může se objevit u každého SOAP prvku.
- Žádný nebo několik atributů daného jmenného prostoru
- Jeden nebo dva potomky[children] v pořadí následujícím:
    1. Volitelná hlavička
    2. Povinné tělo

2.7.2 SOAP hlavička (SOAP header)

Poskytuje mechanismus pro rozšíření SOAP zprávy, obsahující informace jako autentizace, platba a další.

Obsahuje:

- Místní název[local name] hlavičky
- V Jmenném prostoru[namespace name] "http://www.w3.org/2003/05/soap-envelope"
- Žádný nebo několik atributů daného jmenného prostoru
- Žádný nebo několik potomků[children] daného jmenného prostoru
    - Každý potomek SOAP hlavičky je nazýván pomocí Bloku hlavičky SOAP, obsahující vlastní jmenný prostor, potomky a atributy, jako encodingStyle, role, mustUnderstant, relay, které mají také vlastní jmenný prostor a název.

2.7.3 SOAP tělo (SOAP body)

SOAP tělo poskytuje mechanismus pro přenos informací na konečného SOAP příjemce uvedeného v cestě SOAP zprávy (SOAP message path). **Tělo** obsahuje volání a reakce na informace.

Obsahuje:

- Místní název[local name] těla
- V jmenném prostoru[namespace name] "http://www.w3.org/2003/05/soap-envelope"
- Žádný nebo několik atributů daného jmenného prostoru
- Žádný nebo několik potomků[children] daného jmenného prostoru
    - Může mít několik potomků, které mají vlastní jmenný prostor a můžou mít své potomky a atributy, jako encodingStyle.

2.7.4 SOAP porucha (SOAP fault)

SOAP porucha je značka, která obsahuje informace o chybě generované SOAP uzlem. Používá se k označení chybové zprávy, pokud je tento element přítomen, musí být potomkem SOAP těla. Z nějakého důvodu protokol nefunguje správně, v tomto elementu nalezneme informace potřebné k zjištění, proč se tak děje.

3 Knihovny a funkce SOAP

Webové služby v SOAP protokolu jsou zprostředkovány pomocí knihoven, proto jsem sepsal informace k dostupným knihovnám ve vybraných programovacích jazycích a prostředí .NET.

3.1 PHP

Dostupné knihovny pro jazyk PHP je možno vyhledat pomocí vyhledávače Google, zadáme-li vyhledávání "PHP SOAP library". V Prvním odkazu nalezneme odpověď, druhý odkaz obsahuje PHP manuál.

3.1.1 Integrovaná knihovna

Programovací jazyk PHP obsahuje integrovanou knihovnu, pro práci s SOAP protokolem. Tato knihovna procuje s extenzi "libxml", která je součástí PHP od verze 5.0. Knihovna obsahuje vše, co pro tvorbu webové služby pomocí SOAP potřebuji (kromě automatického generování WSDL dokumentu). Podporuje SOAP 1.1, SOAP 1.2 a WSDL 1.1. knihovna může být použita pomocí WSDL nebo non-WSDL režimu. Podporuje posílání SOAP zpráv i SOAP hlaviček a jejich zpracování na SOAP serveru.

3.1.2 NuSOAP

Další knihovnou je NuSOAP obsahující SOAP třídy pro rozšíření integrované knihovny. Nejnovější verze NuSOAP je 0.9.5. Tato verze vyšla 22.6.2011 je možno ji stáhnout ze stránky [8], *Nusoap.sourceforge.net*, © 1999-2009 http://nusoap.sourceforge.net/.

Stáhnutý soubor obsahuje dva adresáře, adresář sample s ukázkami implementace klientů a samotnou knihovnu v adresáři lib.

Umožňují vývojářům vytvářet a využívat webové služby založené na SOAP 1.1, WSDL 1.1 a HTTP 1.0/1.1. Podporuje generování WSDL dokumentu a posílání SOAP zpráv, neobsahuje však modul pro posílání SOAP hlaviček.

3.1.3 Funkce knihovny v PHP

Seznam všech funkcí SOAP knihovny v jazyce PHP se nachází stránce [9], *PHP,* © 2001-2013 http://us.php.net/manual/en/book.soap.php.

- SoapClient::__call — Toto volání SOAP webových metod.
- SoapClient::__soapCall — Ve WSDL modu je SOAP funkce volaná jako metoda objektu SoapClient, ale v non-WSDL nikoli. Může být použita pro posílaní a přijímání SOAP hlaviček.
- SoapClient::__getFunctions — Vrací seznam dostupných SOAP webových služeb popsaných v WSDL dokumentu, tato funkce nemá žádné parametry.
- SoapClient::__getLastRequest — Vrátí poslední SOAP požadavek.
- SoapClient::__getLastResponse — Vrátí poslední SOAP odpověď.
- SoapClient::__setSoapHeaders — Nastaví SOAP hlavičku pro následná volání.

public **SoapClient::SoapClient** ($wsdl [, array $options])

$options může obsahovat parametry "location", "uri", " login", " password", připojení pomoci proxy serveru a další. Parametry " Style" a " use" fungují jen v non-WSDL režimu, v WSDL režim, jsou tyto parametry převzaty z WSDL dokumentu.

Implementace pro WSDL režim

```
$client = new SoapClient("Adresa WSDL dokumentu");
```

Implementace pro non-WSDL režim

```
$client = new SoapClient(null, array( 'location'=>"Adresa Serveru",
                          'uri'=>"Jmenný prostor volané služby"));
```

- SoapServer::addFunction — Přidání jedné nebo více webových metod pro klienta.
- SoapServer::setClass — Nastavuje třídu, která zpracovává SOAP požadavky. Exportuje všechny metody ze zadané třídy.

  public SoapServer::SoapServer ($wsdl [, array $options])

  $options může nastavit verzi SOAP nebo další parametry.

  Implementace pro WSDL režim

```
$server = new SoapServer("Adresa WSDL dokumentu");
```

Implementace pro non-WSDL režim

```
$server = new SoapServer(null, array('uri' => "Adresa Serveru"));
```

- SoapHeader::SoapHeader — Vytvoří nový objekt SoapHeader.

```
$header = new SoapHeader('http://soapinterop.org/echoheader/',
                         'MyHelloRequest',
                         'hello world'));
```

Parametry hlavičky jsou: jmenný prostor SOAP hlavičky, název objektu "SoapHeader", dále následují nepovinné atributy.

3.2 Python

Dostupné knihovny pro webové služby v Pythonu, pomocí SOAP protokolu najdete na této stránce [10], *Python* 29.9.2012 http://wiki.python.org/moin/WebServices.

3.2.1 PySimpleSoap

PySimpleSoap je jednoduchou SOAP knihovnou pro rozhraní webových služeb se strukturou klient a server. Cílem knihovny je jednoduchost a současné podporování nejčastější funkcí. Knihovna je relativně nová a datuje se od začátku roku 2010.

Obsahuje Web2py podporující společnou infrastrukturu k odhalení webových služeb jednoduchým způsobem a to sice pomocí nástrojů služeb (rss, json, jsonrpc, xmlrpc, jsonrpc, amfrpc, amfrpc3).

Knihovna PySimpleSoap podporuje generování WSDL dokumentu posílání SOAP zpráv a hlaviček z SOAP klienta, ale na SOAP serveru zatím však nepodporuje jejich zpracování.

Informace o knihovně PySimpleSoap se nachází na stránce [11], *Pysiimplesoap: Python Simple SOAP Library*, http://code.google.com/p/pysimplesoap/.

3.2.2 ZSI

ZSI je součástí projektu pro vývoj webových služeb poskytující knihovny pro programovací jazyk Python. Zahrnuje klient i server, implementuje protokoly SOAP, WSDL, UDDI, a další.

Knihovnu je nelehké použít, vytváříme-li složité webové aplikace. Knihovna ZSI má pomalý vývoj a poslední verze ZSI 2.0 vyšla na začátku roku 2007.

Obsahuje podrobné posílání SOAP zpráv, nikoli však řádné posílaní SOAP hlaviček. V projektu ZSI najdu i knihovnu s názvem SOAPpy.

Informace o knihovně ZSI se nachází na stránce [12], *Python Web Services,* http://pywebsvcs.sourceforge.net/.

3.2.3 Suds

Knihovna reprezentuje programování v Pythonickém stylu.

Knihovna umožňuje jednoduše vytvořit SOAP klienta pro náročné webové služby používající WSDL dokument, protože se automaticky stará o všechny známé problémy při tvorbě SOAP klienta. Podporuje pouze vytvoření SOAP klienta, nejnovější verze vyšla v roce 2011.

Podporuje nejen posílání SOAP zpráv a vlastních SOAP hlaviček , ale i před definované moduly na autentizaci.

Informace o knihovně Suds se nachází na stránce [13], *Suds,* © 2003-2012, 1.1.2012 https://fedorahosted.org/suds/.

3.2.4 Spyne

Knihovna Spyne je RPC nástroj umožňující při použití několika protokolů snadné odhalení online služeb, které mají přesně definovanou API.

Projekt byl dříve nazýván Rpclib a v současné době podporuje WSDL 1.1, spolu s SOAP 1.1, stejně jako HttpRpc, XmlObject, JsonObject, MessagePackObject a MessagePackRpc protokoly, které lze přepracovat přes HTTP nebo ZeroMQ.

Spyne je framework pro vytváření distribuovaných řešení, která se striktně řídí MVC vzorem, model = spyne.model, View = spyne.protocol a Controller = user kód.

Spyne běží na jakékoliv verzi Pythonu od 2,5 až 2,7.

Knihovna podporuje posílání SOAP zpráv, ale neobsahuje metodu na posílání SOAP hlaviček.

Informace o knihovně Spyne se nachází na stránce [14], *Spyne,* http://spyne.io/.

3.2.5 Soaplib

SOAPlib představuje snadno rozšiřitelnou SOAP knihovnu, která poskytuje několik užitečných nástrojů pro publikaci a vytváření SOAP webových služeb v Pythonu. Knihovna obsahuje generaci WSDL dokumentu, podporu pro komplexní třídní struktury, binární přílohy, jednoduchý framework pro tvorbu dalších serializační mechanismů a knihovnu pro SOAP klienta. Zatím nepodporuje funkčnost s existujícím WSDL dokumentem. Poslední verze vyšla na začátku roku 2011.

Podporuje WSDL 1.1 standard, SOAP 1.1, posílání SOAP hlaviček neobsahuje.

Informace o knihovně Sooaplib se nachází na stránce [15], *GitHub: Soaplib v 2.0 beta documentation,* © 2010 http://soaplib.github.io/soaplib/2_0/.

3.2.6 Ladon

Knihovna Ladon podporuje všestranný přístup pro vytvoření webových služeb.

Pomocí vytvoření jedné služby a její následné vystavení do několika protokolových služeb včetně SOAP 1.1. Knihovna Ladon dynamicky generuje WSDL dokument. Generování WSDL dokumentu umožňuje vystavení typů parametrů pro každou webovou metodu, které jsou definovány prostřednictvím ladonize dekoratéru.

Podporuje webové služby s WSDL dokumentem pro protokol SOAP 1.1, ve verzi Python 2, nebo Python 3.

Obsahuje posílání SOAP zpráv, ale neobsahuje posílání SOAP hlaviček.

Informace o knihovně Ladon se nachází na stránce [16], *Ladon Webservice*, http://ladonize.org/index.php/Main_About

3.2.7 SOAPy

SOAPy je SOAP/XML schematická knihovna pro Python, který používá WSDL a SDL dokument k mapování SOAP služeb založených na API. Pomocí knihovny SOAPy jsou webové služby v Pythonovské aplikaci velice transparentní. Knihovna je navržena pro WSDL 1.0 a SOAP 1.1.

Podporuje posílání SOAP zpráv i hlaviček a jejich zpracování na SOAP serveru, proto byla tato knihovna v oblasti SOAP webových služeb nejlepší. Nadále už ale není udržována. Knihovna je nefunkční od verzi Python 2.5.

Informace o knihovně ZSI se nachází na stránce [17], Adama Elmana, *SOAPy*, 26.4.2001 http://soapy.sourceforge.net/.

3.3 Ruby

Programovací jazyk Ruby obsahuje několik knihoven na zpracování SOAP webových služeb, ty jsou však jen zřídka kdy použitelné. Knihovny se v Ruby nazývají gemy, s kterými Ruby pracuje.

Seznam knihoven najdete na stránkách: [18],[19]

The Ruby toolbox https://www.ruby-toolbox.com/categories/soap,

RubyGems.org: your community gem host, 1.7.2009 https://rubygems.org/

3.3.1 Soap4r

Knihovna Soap4r implementuje SOAP 1.1 a WSDL 1.1 pro Ruby. Tento projekt je uzavřen, ačkoli se stále používá. Knihovna je nekompatibilní s Ruby od verze 1.9, ale jsou pro ní vytvořené úpravy, aby pracovala i v nových verzích Ruby. Bohužel úpravy zatím nefungují v nejnovější verzi Ruby 2.0., zprostředkovává je knihovna Mumboe-soap4r, nebo knihovna soap4r-ruby1.9.

První verze Soap4r vyšla v červenci 2000, poslední verze Soap4r 1.5.8 vyšla 23 září 2007.

Soap4r podporuje dva různé typy serverů, např. StandAlone server a CGI/FastCGI, což je zastaralý standard pro vytvoření dynamické webové stránky.

Existují další knihovny, které jsou na Soap4r založené, jako SimpleWS, který knihovnu Soap4r značně zjednodušuje.

Knihovna Soap4r podporuje posílání SOAP zpráv, ale SOAP hlavičky jsou nefunkční. Nefunkčnost SOAP hlaviček je dána špatnou kompatibilitou s novějším Ruby.

Knihovna v současné době nemá oficiální dokumentaci, která byla stažena po uzavření Soap4r projektu.

Informace o knihovně se nachází na stránce [20], *Tutorialspoint: Simple Easy Learning,* © 2013 http://www.tutorialspoint.com/ruby/ruby_web_services.htm a [21], *Brendonwilson.com,* 2.4.2006 http://www.brendonwilson.com/blog/2006/04/02/ruby-soap4r-wsdl-hell/.

3.3.2 Savon

Savon je knihovna pro vytváření SOAP klientů v programovacím jazyce Ruby. Byla vytvořena, aby poskytovala lehkou a snadno použitelnou alternativu k knihovně soap4r.

Implementace je detailně rozepsaná na oficiálních stránkách, které budete potřebovat, jestli chcete vědět, jak správně volat metody vaší SOAP služby. Knihovna Savon umožňuje velice hodně konfigurací SOAP klienta.

Jedná se o nejpoužívanější knihovnu v programovacím jazyce Ruby obsahujícím pouze SOAP klienta, pomocí WSDL dokumentu, nebo bez použití WSDL dokumentu.

Podporuje posílání SOAP zpráv i SOAP hlaviček. Tato knihovna obsahuje předdefinované SOAP hlavičky týkající se autentizace.

Informace o knihovně Savon se nachází na stránce [22], *Savon: Heavy metal SOAP klient,* 6.10.2009 http://savonrb.com/version1.html.

3.3.3 Savon HandSoap

HandSoap je knihovna pro vytváření SOAP klientů v programovacím jazyce Ruby. HandSoap byla vytvořena, jako alternativou ke knihovně Soap4r. Implementace knihovny HandSoap je více podobná kódu Ruby.

Knihovna se snaží minimalizovat přístup k webovým službám, místo plné abstraktní vrstvy. Tento princip můžu přirovnat k používání nástrojů, pomocí kterých je možné napsat SOAP vazby.

Knihovna umožňuje ladění a opravu chyb na protokolové úrovni. Podporuje WSDL dokument, SOAP verzi 1.1 a 1.2.

HandSoap je mnohem rychlejší, než knihovna Soap4r, protože používá knihovny nízké úrovně pro XML parsování a HTTP komunikaci.

Informace o knihovně HandSoap se nachází na stránce [23], Unwire A/S, *GitHub,* 1.4.2009 https://github.com/unwire/handsoap.

Zajímavé porovnání knihovny Savon s knihovnou HandSoap najdete na stránce

HandSoap vs Savon. http://blog.nofail.de/2010/01/savon-handsoap-shootout/.

3.3.4 WashOut

WashOut je knihovna, která výrazně zjednodušuje tvorbu SOAP webových služeb. Koncový uživatel v knihovně WashOut je jednoduchý Rails kontroler, který obsahuje modul WashOut. Každá SOAP akce koresponduje s určitou metodou v kontroleru. Mapování webových metod a definice parametrů je definováno pomocí metod SOAP akcí.

Knihovna podporuje posílaná SOAP zpráv, ale neobsahuje SOAP hlavičky. Použití SOAP hlaviček by muselo být doprogramováno přímo v projektu Rails.

Informace o knihovně WashOut se nachází na stránce [24], Boris Staal, Peter Zotov. *GitHub,* 15.8.2011 https://github.com/inossidabile/wash_out.

3.4 .NET

Prostředí .NET nepotřebuje instalaci žádné knihovny, funkce pro SOAP webových služeb jsou integrované.

Webová služba v prostředí .NET na SOAP serveru automaticky vytváří WSDL dokument.

Prostředí .NET má na výběr mezi použitím přidání webové reference nebo přidání service reference

Přidání webové reference je založeno na Wsdl.exe a může být použito k vytvoření proxy pro .NET 1.1 nebo 2.0 klienta.

Přidání service reference je založeno na SvcUtil.exe a také vytváří klientům proxy a přídavné Web.config záznamy. Tyto proxy mohou být konzumovány pomocí klientů v .NET 3.0+.

Web Services Description Language Tool (Wsdl.exe) generuje kód pro, XML webovým službám a XML klientům webovým služeb z WSDL dokumentu nebo XSD schémat a discavery protokolu. Podrobnější informace najdete na stránce [25], *Msdn: Visual Studio,* © 2013 http://msdn.microsoft.com/en-us/library/vstudio/7h3ystb6(v=vs.100).aspx.

ServiceModel Metadata Utility Tool (Svcutil.exe) je nástroj pro generování servisního modelu kódu z dokumentů metadat a obráceně. Svcutil.exe může generovat kód pro zakázky služeb, klienty a datové typy z dokumentů metadat. Tyto dokumenty metadata mohou být trvale uskladněny, nebo získány online. Online vyhledávání následuje buď WS-Metadata Exchange protokol nebo protokol DISCO. Podrobnější informace možno dohledat na stránce [26], *Msdn*, 8.2.2012 http://msdn.microsoft.com/en-us/library/aa347733.aspx.

4 Implementace SOAP protokolu.

Předchozí kapitoly popisují WSDL dokument, SOAP protokol a knihovny. Příklady implementace jsem začal psát postupně v jednotlivých jazycích. Pro zprovoznění kódů v této kapitole nainstalujte vše potřebné, instrukce najdete v manuálu, který naleznete v příloze.

Implementace je provedená v Ruby 1.93, Pythonu 2.7, PHP 5.3.8 a Microsoft Visual studiu 2010, pro práci v prostředí .NET, na platformě Microsoft Windows 7.

V následující části práce se pokusím o vytvoření vlastní SOAP webové služby a začnu v prostředí .NET.

4.1 .NET

Ukázka je napsaná v programu Visulal studio 2010 v jazyce C#. Inspiraci potřebnou k zpracování úkolu jsem nalezl na stránce zabývající se vytvořením jednoduché WCF služby [27], *John Been.* 17.5.2009 http://johnwsaunders3.wordpress.com/2009/05/17/how-to-consume-a-web-service/.

Vytvořil jsem ve Visulal studiu 2010 prázdný asp.net projekt. Následně jsem tomuto projektu přidal novou webovou službu, tedy soubor s příponou .asmx, který bude obsahovat funkčnost webové služby.

Server

```
public class jeNazevStatu : WebService
{
      [WebMethod]
      public bool JeNazevStatu(string s)
      …

      [WebMethod]
      public int Dvakrat(int value)
      …

      [WebMethod]
      public int Secti(int a, int b)
      …

      [WebMethod]
      public int Nazdar()
      …

      [WebMethod]
      public int Ahoj(string kdo)
      …
}
```

Metoda JeNazevStatu kontroluje, zda je získaný parametr uveden v seznamu státu.

Metoda Dvakrat vrací dvojnásobek uvedeného parametru.

Metoda Secti vrací součet dvou uvedených čísel.

Methoda Nazdar vypíše pozdrav.

Metoda Ahoj pozdraví osobu, jejíž jméno je předávané parametrem.

Webová metoda se vytváří stejným způsobem, jako běžná metoda v jazyce C#. Před deklaraci metody je tedy nutno napsat [WebMethod].

Dále služba obsahuje posílání hlaviček, které jsem našel na stránce [29], Ahmed Shokr. *Code Project: For those who code,* 29.6.2008 http://www.codeproject.com/Articles/27365/Authenticate-NET-Web-Service-with-Custom-SOAP-Head.

```
public UsernameToken zakaznik;

public class UsernameToken : SoapHeader
…

[WebMethod]
[SoapHeader("zakaznik", Required = true)]
public string GetBalance()
…

private bool kontrolaZakaznik()
…
```

Vlastní SOAP hlavičku jsem vytvořil pomocí třídy s názvem "UsernameToken", které dědí z "System.Web.Services.Protocols.SoapHeader". Správné posílání SOAP hlavičky by mělo dodržovat takto určený tvar. "UsernameToken" obsahuje uživatelské jméno a heslo.

Následně napíši webovou metodu, která tuto hlavičku bude reprezentovat výsledkem autentizace.

Nezapomeňte před vytvářením SOAP klienta webovou službu zkompilovat. Pravým tlačítkem na webovou službu a kliknout na tlačítko "build".

Po spuštění SOAP serveru můžu na jeho adrese najít vygenerovaný WSDL dokument a strukturu SOAP požadavků a odpovědi.

Klient

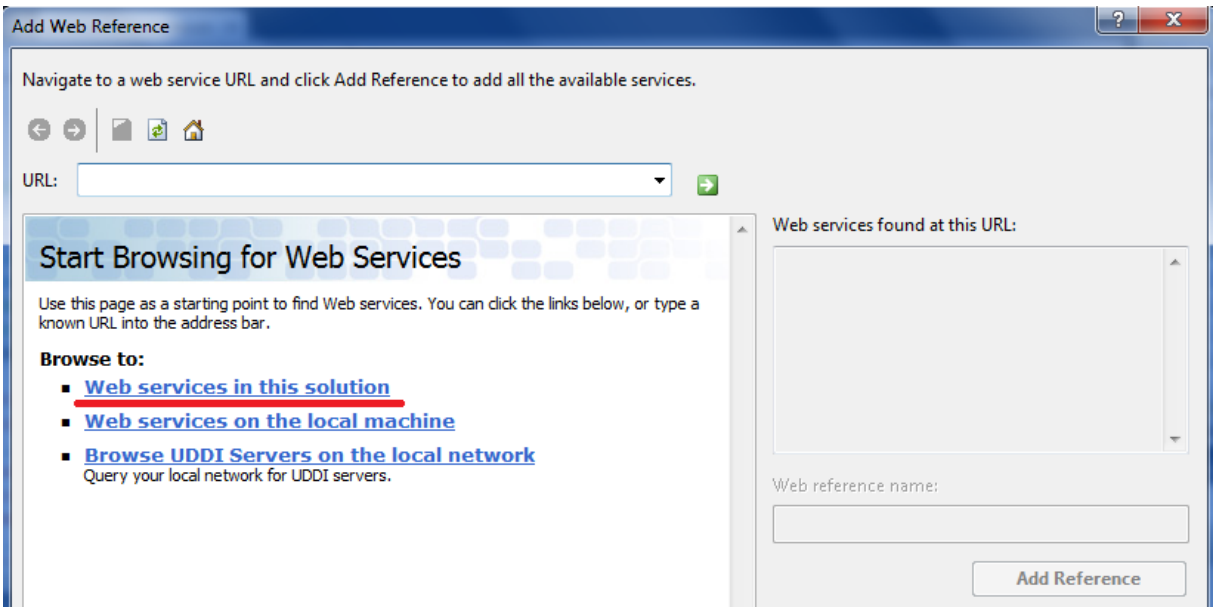
Sepsal jsem podrobný postup pro vytváření klienta v prostředí .NET, neboť souvisí s textem v dalších kapitolách.

Visual studio umožňuje vytvoření konzumaci webové služby jako desktopovou aplikaci nebo webovou aplikace. Použil jsem aplikaci webovou, jelikož na internetu je možné najít většinou jen tutoriály s desktopovou aplikací.

Klíčovým bodem konzumace webové služby je přidání reference. Visual studio 2008 a 2010 již nemají "Add Web Reference" přímo v menu, jak tomu bylo ve Visual studiu 2005. Menu obsahuje možnost přidání "Service reference“. Výběr této možnosti vygeneruje WCF v C# proxy klienta, ale já použiji webovou službu asmx proxy klienta. Proto se budu muset dostat k dialogovému oknu "Add Web Reference", který je vnořen v dialogovém okně "Add Servis Reference".

Postup:

1. Kliknu pravím tlačítkem myši na Reference a vyberu "Add Service Reference"
2. Kliknu na tlačítko "Advanced" (pokročilé)
3. Kliknu na tlačítko "Add Web Reference"
4. Kliknu na tlačítko "Web services in solution" , které zobrazí všechny webové služby v mém projektu



Obrázek 4.1: Services in Solution

Stane-li se, že SOAP server není ve stejném projektu, nebo potřebuji přidat jiný SOAP server, můžu do kolonky "URL" přidat adresu s WSDL dokumentem, nebo SOAP serverem, který WSDL generuje.

Před přidáním reference nesmím zapomenout službu nějak vhodně pojmenovat a pak už jí můžu přidat tlačítkem "Add Reference". Reference může být přidaná ke každé webové službě pouze jednou.

Souboru "Default.aspx" obsahuje volání přidané reference, pomocí této reference je následně možné volání vytvořených webových metod. Přidání je provedeno ve tvaru název reference, název webové služby, která následuje reprezentující název webové služby.

```
serviceNET.jeNazevStatu sluzby = new serviceNET.jeNazevStatu();

Console.Write(Dvojnásobek čísla 5 je " + sluzby.Dvakrat(5));
Console.Write("Výsledek součtu : 4 + 5 = " + sluzby.Secti(4, 5);
Console.Write("Zpráva : " + sluzby.Ahoj("Lucie"));

serviceNET.jeNazevStatu svc = new serviceNET.jeNazevStatu();
if (sluzby.jestat("USA") == true)
     Console.Write("Výsledek : Je Stat");
else
     Console.Write("Výsledek : Neni Stat");

serviceNET.jeNazevStatu svch = new serviceNET.jeNazevStatu();
serviceNET.UsernameToken svcu = new serviceNET.UsernameToken();
svcu.Username = "tom";
svcu.Password = "ric";
svch.UsernameTokenValue = svcu;
Console.Write(svch.GetBalance());
```

Posílanou SOAP hlavičku jsem připojil k SOAP zprávě a zavolám webovou metodu pro její vyhodnocení.

Pokud je nutné volání webové metody pořádně ošetřit, použiji následující ukázku, která vypíše chybu při špatném volání webové služby nebo webové metody.

```
serviceNET.jeNazevStatu svc = null;
bool success = false;
try
{
     svc = new serviceNET.jeNazevStatu();
     if (svc.jestat("USA") == true)
          Console.Write("Výsledek : Je Stat");
```

```
      else
            Console.Write("Výsledek : Neni Stat");
      success = true;
}
finally
{
      if (!success && svc != null)
      {
      Console.Write("Nesprávné připojení .");
      }
}
```

4.1.1 Připojení na SOAP server v PHP

Připojení k SOAP serveru v jazyce PHP je velice jednoduché a totožné s předchozí ukázkou SOAP klienta v prostředí .NET, protože při vytváření SOAP serveru v jazyce PHP jsem dával pozor na kompatibilitu s prostředím .NET.

Stačí přidat referenci podle popsaného postupu. Jediným rozdílem je název reference, který je při přidávání libovolný.

Následující kód v souboru Default.aspx volá metodu s parametrem v námi vytvořené referenci. Vytvořím objekt s webovou službou, který následně použiji pro volání webové metody.

```
serviceIntegrovana.SomeService sluzba = new
      serviceIntegrovana.SomeService();
Console.Write("Dvojnásobek čísla 5 je " + sluzba.Dvakrat(5));
```

Komunikace s knihovou NuSOAP v jazyce PHP je bez problémů, protože tato knihovna generuje WSDL dokument vyhovující .NET požadavkům.

```
serviceNuSOAP.SimpleService sluzba_nusoap = new
      serviceNuSOAP.SimpleService();

Console.Write("Výsledek : " + sluzba_nusoap.Ahoj("Lucie"));
```

4.1.2 Připojení na SOAP server v Pythonu

Implementace SOAP klienta je opět totožná, protože použitá knihovna PySimpleSoap v jazyce Pythonu je otestována s funkčností připojení na webové služby v prostředí .NET a jiné.

```
servicePython.PythonSluzbyService sluzby = new
      servicePython.PythonSluzbyService();
Console.Write(""Zpráva : " + sluzby.Ahoj("Lucie"));
```

4.1.3 Připojení na SOAP server v Ruby

Použitá knihovna soap4r v režimu "Stand alone Server negeneruje WSDL dokument a nepodporuje metadata, podle kterých by bylo možné webovou službu v prostředí .NET určit. Tento nedostatek neumožní přidat referenci, proto jsem se v prostředí .NET na tento SOAP server nemohl připojit.

Jestliže použiji knihovnu soap4r s WSDL dokumentem, mohu pomocí WSDL dokumentu na tento SOAP server přidat referenci. Bohužel přidání není kompletní a nebudou zaregistrovány žádné metody, protože prostředí .NET očekává jiný formát WSDL dokumentu.

4.2 PHP

Moje první implementace SOAP v jazyce PHP byla vytvořena podle tutoriálu na stránce [28], Dmitry Stogov, Zend Developer Zone, 30.11.2001 http://devzone.zend.com/25/php-soap-extension/. Autor této stránky je autorem SOAP rozšíření v PHP. Článek na této stránce obsahuje přehledně popsanou implementaci SOAP v jazyce PHP, proto jsem vyzkoušel všechny příklady z této stránky a narazil jsem na chybu.

Fatal error: Class 'SoapClient' not found

Jazyk PHP nezná třídu SoapClient, integrovaná knihovna pro SOAP nebyla v jazyce PHP zapnuta. Na internetu jsem vyhledal způsob, jak SOAP knihovnu zapnout.[1]

Zapnutí knihovny provedu odkomentováním řádku s extension=php_soap.dll v souboru php.ini řádek. Soubor php.ini obsahuje konfiguraci jazyka PHP.

Vymazal jsem středník a restartoval službu Wamp, ale chyba pořád přetrvávala. Po neúspěšných hodinách hledání, zda nainstalovaný Wamp obsahuje soubor php_soap.dll. Prohlížel jsem také funkci phpinfo(), zkoušel jsem EasyPhp, reinstaloval Wamp a instaloval jiné verze.

Soubor php.ini můžu najít také v Apachi HTTP Serveru, je v adresáři wamp\bin\apathe, kde je nutné odkomentovat extension=php_soap.dll. Jazyk PHP se podle tohoto zjištění řídí právě tímto souborem a nejedná se o soubor php.ini v adresáři wamp\bin\php.

Následně jsem restartoval službu Wamp a výše zmíněné příklady fungovaly bez problému, kromě prvního a druhého klienta, protože služby na které se připojují už neexistují.

Příklady SOAP serverů z tohoto tutoriálu jsem zkusil použít ke komunikaci s klientem v prostředí .NET. Přidání reference bylo umožněno, ale bohužel její volání se nezdařilo, protože se v jmenných prostorech projektu nezobrazila. Důvod je zřejmě starší verze WSDL dokument, a .NET nedovedl rozpoznat získanou službu, i když jí dokázal přidat.

Zkoušel jsem další možnosti, jako například SOAP server v non-WSDL modu, ale v prostředí .NET jsem takhle nemohl přidat referenci, protože PHP server nevypisoval žádný WSDL dokument, nebo jinou alternativu a knihovny v prostředí .NET tento případ nepodporují.

Našel jsem už hotový projekt zabývajícím se kompatibilitou jazyka PHP v prostředí.NET. Podle tohoto projektu jsem při implementaci postupoval. Tento projekt je zveřejněn na stránce [30], *Nilzor's Techblog,* 5.6.2010 http://www.nilzorblog.com/2010_05_01_archive.html.

---

[1] http://stackoverflow.com/questions/6760716/wamp-stack-php-fatal-error-class-soapclient-not-found http://stackoverflow.com/ stránka obsahující dotazy na různé problémy týkající se programování.

SOAP server v jazyce PHP vypadal takto:

Server_Integrovana.php

```
ini_set("soap.wsdl_cache_enabled", "0");
```

Tento řádek by změnil nastavení SOAP knihovny v PHP pro tuto instanci, ale já používám defaultní nastavení, proto nemusí být v SOAP serveru obsaženo.

```
function Dvakrat($valueObj)
{
      //rozbalí vstupní data
      $valueArr = get_object_vars($valueObj);
      $value = $valueArr["value"];
      $result = $value * 2;
      // Zabalí a vrátí výsledek
      return array("DvakratResult" => $result);;
}
```

Vytvořil jsem webovou metodu s názvem Dvakrát, která posílá data jako objekt, protože prostředí .NET požaduje posílání parametrů jako objekty, shodujícími se s použitým WSDL dokumentem. WSDL dokument by měl obsahovat stejný jmenný prostor, jaký generuje .NET, aby nedošlo k problémům s SOAP klientem v prostředí .NET.

Metoda Dvakrat vynásobí dvěma přijatý parametr.

WSDL dokument určuje kódování SOAP webové služby. Kódování najdu v elementu body pod SOAPAction, nejčastější používané kódování je literal. Pro komunikaci s prostředím NET je vhodné požívat v WSDL dokumentu právě kódování literál. WSDL dokument generovaný v prostředí .NET používá právě kódování literal. Použitý WSDL dokumenty najdu v adresáři Server\ PHP\PHP_Integrovana s názvem DvakratService.wsdl.

WSDL dokument připojíme na SOAP server pomocí odkazu na SOAP serveru. Odkaz na SOAP server do location, který najdu na konci dokumentu v elementu service pod elementem port.

```
$server = new SoapServer("DvakratService.wsdl");
$server->addFunction("Dvakrat");
$server->handle();
```

Následně připojím SOAP server pomocí WSDL dokumentu a přidám metodu Dvakrat, kterou mám napsanou výše, nakonec aktivuji SOAP server pomoci metody handle().

Client_Integrovana.php

```
$client = new SoapClient("DvakratService.wsdl", array('trace'=> 1));
```

Připojíme se na SOAP server pomocí WSDL dokumentu, jehož název je napsaný v prvním parametru. Další parametr předávaný v poli, podle specifikace v PHP manuálu, umožňuje zapnout sledování SOAP požadavků a odpovědí.

```
var_dump($client->__getFunctions());
```

Metoda getFunction() vypíše všechny dostupné webové metody, ale funguje jenom v případě použití WSDL dokumentu.

```
$vysledek = $client->Dvakrat("5")->DvakratResult;
```

Volání webové metody v jazyce PHP jsem zatím prováděl jako přístup k třídě třídě (viz stránka [28], Dmitry Stogov, Zend Developer Zone, 30.11.2001 http://devzone.zend.com/25/php-soap-extension/). Tento způsob nebyl pro můj SOAP server nefunkční.[2]

Začal jsem tedy hledat na internetu řešení a narazil jsem na tuto stránku [31], *Coding Friends*, 16.4.2010 http://www.codingfriends.com/index.php/2010/04/16/soap-client-calling-net-web-service/.

Chyba není v předávání parametrů webové metody, ale ve formátu návratové hodnoty objektu $client. Webová metoda napsaná v SOAP serveru, předává objekt, protože je napsaná s ohledem na kompatibilitu s prostředím .NET.

---

[2] V podobě : $vysledek = $client->Dvakrat($parametrs), ale dostanu další chybu.

error: Object of class stdClass could not be converted to string

Protože jsem nevěděl co tento error značí, zkoušel jsem jiné volání, které najdeme v PHP manuálu v sekci SOAP. Nejprve jsem napsal zastaralé volání $vysledek = $client->__Call ("Dvakrat", array("cislo" => $parametr)). Která předává parametr pomocí pole, tento způsob volání jsem se dozvěděl v komentářích na stránce PHP manuálu.

Dále jsem vyzkoušel $vysledek = $client->__soapCall("Dvakrat", array("cislo" => $parametrs)) a pořád jsem nic nevyřešil.

Abych získal přístup ke skutečnému výsledku, je nutné odkázat se na první část vraceného pole s názvem DvakratResult, nejedná se o DvakratResponse z WSDL dokumentu.

Následně byla zobrazena další chyba, která se teď týkala předávaného parametru. Předávané parametry posílaných webových metod musí mít totožný název, jako ve vytvořeném SOAP serveru. Použiji volání pomocí soapCall() nebo Call(), protože moje metoda musí byt volaná jako objekt. Výsledné volání bude vypadat takto:

```
$vysledek = $client->__soapCall("Dvakrat", array(array("value" =>
      "5")))->DvakratResult;
```

nebo

```
$vysledek = $client->Dvakrat(array("value" => "5"))->DvakratResult;
```

nebo

```
$vysledek = $client->Dvakrat(array("value" =>"5")))->DvakratResult;
```

```
print("Vysledek je : ".$vysledek );
print ("<br /><br />");
print ("REQUEST:" . htmlentities($client->__getLastRequest()));
print ("<br /><br />");
print ("RESPONSE:" . htmlentities($client->__getLastResponse()));
```

Vypsání výsledku je jednoduché, použiji echo, nebo print zvolené proměnné, do které jsem uložil volání webové metody, v mém případě je to proměnná `$vysledek`. Pro funkčnost getLastRequest() metody musí být při připojování se k SOAP serveru zapnutý trace, potom je možné vypsání SOAP požadavku a odpovědi.

SOAP požadavek.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
      xmlns:SOAPENV=http://schemas.xmlsoap.org/soap/envelope/
      xmlns:ns1="http://myservice.org/">
  <SOAP-ENV:Body>
      <ns1:Dvakrat>
            <ns1:value>5</ns1:value>
      </ns1:Dvakrat>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Odpověď

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
      xmlns:ns1="http://myservice.org/">
  <SOAP-ENV:Body>
      <ns1:DvakratResponse>
            <ns1:DvakratResult>10</ns1:DvakratResult>
      </ns1:DvakratResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4.2.1 SOAP hlavička v PHP

Zkoušel jsem poslat hlavičku v tutoriálu ze stránky [28], Dmitry Stogov, Zend Developer Zone. 30.11.2001. http://devzone.zend.com/25/php-soap-extension/, kde se mi povedlo poslat hlavičku z SOAP klienta na SOAP server, pomocí metody setSoapHeaders(). [3] Bohužel jsem ještě nevěděl, jak na straně SOAP serveru hlavičku přijmout a vyhodnotit.

V PHP manuálu – konkrétně na stránce „sekce věnované SOAP serveru“, jsem dole v komentářích našel implementaci SOAP hlavičky. [4]

Implementací tohoto příkladu se podobá následující ukázce.

Server_Integrovana_Header.php

```
$server = new SOAPServer(null, array('uri'=>'urn:MySoapService'));
$server->setClass('mysoapclass');
$server->handle();
```

SOAP server v jazyce PHP nemá metodu pro čtení SOAP hlavičky, proto je nutné provést tuto akci pomocí metody setClass(), která mapuje třídu.

Zapínáme SOAP server v non-WSDL režimu. Metoda setClass() obsahuje webové metody a čte poslanou hlavičku, protože není možné použít metody addFunction() a setClass() zároveň. Důvodem této implementace pomocí jedné z těchto metod je jejich samostatné vytváření XML.

---

[3] V podobě: `$client->__soapCall("getQuote", array($a), NULL, new SoapHeader('getQuote','getQuoteRequest','hello world'));`

[4] PHP manuál. http://us3.php.net/manual/en/soapserver.soapserver.php#78872.

```
class mysoapclass
{
      public function UsernameToken( $username, $password )
      …
```

Metodu pro čtení SOAP hlaviček musím pojmenovat stejným názvem jako název poslané hlavičky. Pakliže je pravidlo dodrženo, SOAP hlavička se automaticky načte a vloží se parametry heslo a uživatelské jméno. Tuto informaci jsem se dozvěděl z internetové stránky [32], *Stackoverflow*. 22.12.2009 http://stackoverflow.com/questions/1948897/process-soap-headers-in-php, protože v PHP manuálu nebyla metoda setClass() podrobně specifikována.

```
      function checkConsumer()
      …

      public function Dvakrat($value)
      …

}
```

Vytvořím webovou metodu pro vyhodnocení obdržené hlavičky a implementuji webovou metodu naší služby, která vrací dvojnásobek získaného parametru.

Další užitečné příklady SOAP hlaviček se nachází na stránkách[5]: [33],[34]

Handbook http://www.bitpapers.com/2012/04/php-using-authentication-with-soap_09.html a http://www.bitpapers.com/2012/04/php-generating-soap-header.html

Client_Integrovana_Header.php

```
$opts=array('location'=>
      'http://localhost/PHP_SOAP/phpServer/SomeServiceH.php','uri' =>
      'urn:MySoapService','trace' => 1 );
$client = new SoapClient(null, $opts);
```

Připojení na SOAP server v non-WSDL režimu.

```
class SoapHeaderUsernameToken
```

[5] Zkoušel jsem podle těchto stránek přidat SOAP hlavičky ale vyskytla se chyba v posílání XML.

```
{
    public $Password;
    public $Username;

    public function __construct($l, $p)
    …
}
```

Naimplementuji třídu pro vytvoření vlastního formátu posílané SOAP hlavičky, kde je uživatelské jméno a heslo zabalené v elementu "UsernameToken".

```
$wsu = 'urn:MySoapService';
$usernameToken = new SoapHeaderUsernameToken('tom', 'ric');
$soapHeaders[]= new SoapHeader($wsu,'UsernameToken',$usernameToken);
```

Jednoduše vytvářím SoapHeader, do kterého vkládám název jmenného prostoru a data posílané SOAP hlavičkou.

```
$client->__setSoapHeaders($soapHeaders);
```

Posílám SOAP hlavičku v předem definovaném formátu. SOAP hlavička bude připojena k první posílané SOAP zprávě. [6]

```
try
{
     $vysledekHeader = $client->checkConsumer();
     $vysledek= $client->__soapCall("Dvakrat",array("value" =>"5"));
}
catch (Exception $v)
…

print("Vysledek je : ".$vysledek );
print($vysledekHeader);
```

Volám webovou metodu a kontrolu poslané SOAP hlavičky. Nakonec vypíšu výsledek a metoda getLastRequest() získá celý SOAP požadavek.

[6] `<SOAP-ENV:Header><ns2:UsernameToken>`
`<Password>ric</Password><Username>tome</Username>`
`</ns2:UsernameToken></SOAP-ENV:Header>`

4.2.2 Webová služba přes NuSOAP

Integrovaná knihovna v jazyce PHP má velikou nevýhodu, nepodporuje generování WSDL dokumentu. Z tohoto důvodu jsem naprogramoval webovou službu používající NuSOAP.

Stáhnu si NuSOAP ze stránky SourceForge.net. http://sourceforge.net/projects/nusoap/, balíček obsahující příklady a adresář lib, protože ve stáhnutém balíčku můžu naleznout příklady pouze s SOAP klientem, je nutné na internetu pohledat implementaci SOAP server. Tutoriál pro zprovoznění NuSOAP webové služby můžete najít na stránce [35], *Sanity Free Coding,* 3.7.2006 http://www.sanity-free.org/125/php_webservices_and_csharp_dotnet_soap_clients.html.

Adresář lib ze staženého balíčku NuSOAP vkládám do adresáře, ve kterém mám uloženou webovou službu, abych jí mohl jednoduše v PHP kódě připojit.

Server_NuSoap.php

```
require_once("lib/nusoap.php");
```

Připojení k souboru nusoap.php, který je součástí NuSOAP balíčku v adresáři lib, tento řádek nám umožní používat knihovnu NuSOAP.

```
function Ahoj($kdo)
…

function Dvakrat($valuee)
…

function Secti($a,$b)
…
```

Vytvoření webových metody.

```
$server = new soap_server();
$server->configureWSDL("SimpleService");
```

Vytvoření SOAP serveru pomocí metody soap_server() je velmi jednoduché. Metoda configureWSDL umožňuje generování WSDL dokumentu a její parametr představuje název vytvářené webové služby. Tato metoda je praktická, protože nemusím WSDL dokument psát, hledat na internetu, nebo ho vytvářet pomocí WSDL generátoru.

```
$namespace = "http://example.com/sample.wsdl";
$server->register('Hello_to_you',
                array('name'=>'xsd:string'),
                array('return'=>'xsd:string'),
```

```
                $namespace,
                false,
                'rpc',
                'encoded',
                'A simple Hello World web method');
```

Registraci webové metody musím napsat se všemi parametry, protože budu chtít tento SOAP server poskytovat dalším jazykům. U webové metody v tomto tvaru mohu lehce kontrolovat, jedná se o její volání a přesné definice parametru.

```
$POST_DATA = isset($GLOBALS['HTTP_RAW_POST_DATA'])
                ? $GLOBALS['HTTP_RAW_POST_DATA'] : '';
$server->service($POST_DATA);
```

Důležitá část, která zobrazuje data na SOAP serveru a určuje, v jakém tvaru se nacházejí. Následně zapnu SOAP server a vložím do něj moje získané data.

Jako SOAP klienta na SOAP server v NuSOAP použiju integrovanou knihovnu v jazyce PHP.

Client_NuSoap.PHP

```
$client=new soapclient('Adresa k server.php?wsdl,array('trace'=>1));
```

Připojí se na SOAP server pomocí WSDL dokumentu vygenerovaným na SOAP serveru v NuSOAP. WSDL dokument naleznu na adrese SOAP serveru, ke které přidám "?wsdl".

```
$vysledekAhoj = $client->__soapCall('Ahoj',array('kdo' => 'Lucie'));
$vysledekDvakrat= $client->__soapCall('Dvakrat',array('value'=> 5));
$vysledekSecti= $client->__soapCall('Secti',array('a'=> 5,'b'=> 4));
```

Volání webové metody pomocí metody soapCall(). Následně je možné vypsat výsledky a posílaný SOAP požadavek.

NuSOAP neobsahuje posílání SOAP hlaviček, pro jejich implementaci bych je musel posílat pomocí xml, tedy celý výraz připojit k posílané SOAP zprávě. Zajímavý příklad o posílání SOAP hlaviček v NuSOAP jsem našel na stránce [35], *Adobe Developer Connection,* © 1996-2011 https://developer.omniture.com/en_US/documentation/omniture-administration/c-api-admin-sample.

4.2.3 Připojení na SOAP server v prostředí .NET

Opět používám SOAP klienta pomocí integrované knihovny jazyce PHP. Při vytváření SOAP klienta jsem si musel dávat pozor na názvy posílaných parametrů, protože se museli shodovat s názvy v SOAP serveru, jak je určeno ve WSDL dokumentu. Rovněž je nutné kontrolovat přístup k výsledkům pomocí odkázání se na vráceného pole s názvem jestatResult. Důvody jsem vysvětloval při tvorbě prvního SOAP klienta v jazyce PHP.

Client_NET.php

```
$client=new
SoapClient("http://localhost:57478/jeNazevStatu.asmx?wsdl");

$vysledek = $client->jestat(array("s" => "USA"))->jestatResult;
$vysledekNazdar = $client->Nazdar()->NazdarResult;
$vysledekAhoj = $client->Ahoj(array("kdo" => "Lucie"))->AhojResult;
$vysledekSecti = $client->__soapCall("Secti", array(array('a'=> 3,
     'b'=> 3)))->SectiResult;
$vysledekDvakrat = $client->Dvakrat(array("value" => 5))
     ->DvakratResult;
```

Posílání SOAP hlaviček na SOAP server v prostředí .NET, proběhne stejným způsobem, který jsem v kapitole SOAP hlavička v PHP už zmínil. Jediná změn je v jmenném prostoru.

```
  class SoapHeaderUsernameToken
  {
    public $Username;
    public $Password;

    public function __construct($l, $p)
    …
  }

$namespace = 'http://tempuri.org';
$usernameToken = new SoapHeaderUsernameToken('tom', 'ric');
$soapHeaders[] = new SoapHeader($namespace, 'UsernameToken',
                                    $usernameToken);
$client->__setSoapHeaders($soapHeaders);
$vysledekHeader = $client->GetBalance()->GetBalanceResult;
```

4.2.4 Připojení na SOAP server v Pythonu

Připojení na SOAP klienta v jazyce PHP. Volání webové metody probíhá pomocí pole, protože v Pythonu napsaný SOAP server očekává objekt. Dané parametry musí mít opět stejné názvy, které jsme uvedl v SOAP serveru. Rovněž je nutné kontrolovat přístup k výsledkům pomocí, odkázání

se na vrácené pole s názvem SectiResult. Důvody jsem vysvětloval při tvorbě prvního SOAP klienta v jazyce PHP.

Připojuji se na SOAP server napsaný v jazyce Pythonu pomocí knihovny PySimpleSoap.

Client_python.php

```
$client= new SoapClient("http://localhost:8008/",array('trace'=>1));
var_dump($client->__getFunctions());
```

Připojení na webovou službu pomocí WSDL dokumentu a vypsání všech dostupných funkcí z WSDL dokumentu.

```
$vysledekSecti = $client->__soapCall("Secti",
                 array(array("a" => 2,"b" => 3)))->SectiResult;
$vysledekAhoj = $client->Ahoj(array("kdo" => "Lucie"))->AhojResult;
$vysledekDvakrat=$client->Dvakrat(array("value"=>5))->DvakratResult;
```

Volání webových metod, které následně vypíšu pomocí metody print().

4.2.5 Připojení na SOAP server v Ruby

SOAP klient pro webovou služby implementovanou pomocí knihovny Soap4r za použití WSDL dokumentu. Funkčnost v jazyce PHP je zaručena podporou WSDL dokumentu verze 1.1. Postup je úplně totožný jako v předchozích ukázkách.

Clietn_ruby.php

```
$client = new SoapClient("http://localhost:2000/wsdl/hws.wsdl",
array('trace' => 1));
$vysledek=$client->__soapCall("hello_world",array("kdo"=>"jmeno"));
```

4.3 Python

Pro vytvoření SOAP webové služby v jazyce Pythonu jsem vybral knihovnu PySimpleSoap, protože je relativně nová od roku 2010. Knihovna podporuje generováni WSDL dokumentu a má jednoduchou implementaci. Jedinou nevýhodou je, že nepodporuje zpracování poslané SOAP hlavička na straně serveru. Pomoc při instalaci je sepsána v manuálu přiloženém v příloze.

Server_PySimpleSoap.py

```
from pysimplesoap.server import SoapDispatcher, SOAPHandler
from BaseHTTPServer import HTTPServer
```

Připojení knihovny probíhá pomocí importu tříd, SoapDispatcher tato třída je potřebná pro vytváření SOAP webových služeb, HTTPServer používám při vytváření webového rozhraní a SOAPHandler použité rozhraní vytváří.

```
def secti(a,b):
    return a+b

def ahoj(kdo):
    return 'Ahoj , '+kdo


def dvakrat(value):
    return value*2
```

Naimplementuji funkčnost webových metod jako normální metody v jazyce Python.

```
dispatcher = SoapDispatcher(
    'PythonSluzby',
    location = "http://localhost:8008/",
    action = 'http://localhost:8008/', # SOAPAction
    namespace = "http://example.com/sample.wsdl", prefix="ns0",
    trace = True,
    ns = True)
```

Vytváření SOAP serveru pomocí metody SOAPDispatcheru(), její parametry jsou: název služby, adresa služby, SOAPAction, který určuje volanou webovou metodu. Jmenný prostor služby a trace, který zapíná možnost vidět veškerou komunikaci na SOAP serveru.

```
dispatcher.register_function('Secti', secti,
    returns={'SectiResult': int},
```

```
    args={'a': int,'b': int})

dispatcher.register_function('Ahoj', ahoj,
    returns={'AhojResult': str},
    args={'kdo': str})

dispatcher.register_function('Dvakrat', dvakrat,
    returns={'DvakratResult': int},
    args={'value': int})
```

Provedení registrace webových metod probíhá pomocí metody register_function(), parametry této metody jsou: název výstupní webové metody, název implementované metody. Metoda obsahuje parametr return, který určuje, co bude webová metoda vracet. Nakonec určuji název a formát vstupního parametru.

```
httpd = HTTPServer(("", 8008), SOAPHandler)
httpd.dispatcher = dispatcher
httpd.serve_forever()
```

Zapnu server na portu 8008 a pomocí metody SoapDispatcher() je na adrese http://localhost:8008/ automaticky vygenerován WSDL dokument.

Client_PySimpleSoap.py

```
from pysimplesoap.client import SoapClient

client = SoapClient(
    location = "http://localhost:8008/",
    action = 'http://localhost:8008/', # SOAPAction
    namespace = "http://example.com/sample.wsdl",
    soap_ns='soap',
    trace = True,
    ns = False)
```

Připojení knihovny a připojení SOAP klienta na SOAP server bez použití WSDL dokumentu k připojení je možné použít WSDL dokument, ale změnil by se kód na získání výsledku volané webové metody. [7] Při použití připojení pomocí WSDL dokumentu je nutné přistupovat k výsledku webových metod jako k objektu ve tvaru ['AhojResult]'. [8]

[7] Připojení s WSDL: `client = SoapClient(wsdl="http://localhost:8008/")`
[8] `vysledekSecti = responseSecti['SectiResult']`

```
client['AuthHeaderElement'] = {'username': 'tom', 'password': 'ric'}
```

Posílání SOAP hlavičky s uživatelským jménem a heslem zabaleným do 'AuthHeaderElement'. [9] V dokumentaci knihovny PySimpleSoap jsem našel několik způsobu posílání SOAP hlaviček.

```
odpovedSecti = client.Secti(a=1, b=2)
vysledekSecti = odpovedSecti.SectiResult
print int(vysledekSecti)

odpovedAhoj = client.Ahoj(kdo="Lucie")
vysledekAhoj = odpovedAhoj.AhojResult
print vysledekAhoj

odpovedDvakrat= client.Dvakrat(value=10)
vysledekDvakrat = odpovedDvakrat.DvakratResult
print "vysledek : 10 * 2 = ",vysledekDvakrat
```

Volání webových metod a získání výsledku pro jejich vypsání. Názvy parametrů při volání webové metody musí být totožné s názvy, jaké jsme použili v SOAP serveru - v SOAP serveru jsem je už přesněji definoval.

SOAP požadavek

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
      xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Header>
      <AuthHeaderElement>
            <username>tom</username>
            <password>ric</password>
      </AuthHeaderElement>
  </soap:Header>
  <soap:Body>
      <Dvakrat xmlns="http://example.com/sample.wsdl">
            <value>10</value>
      </Dvakrat>
  </soap:Body>
</soap:Envelope>
```

[9] Záhlaví ve tvaru : <soap:Header><AuthHeaderElement><username>tom</username>
<password>ric</password></AuthHeaderElement></soap:Header>

SOAP odpověď

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
      xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance">
  <soap:Body>
      <DvakratResponse xmlns="http://example.com/sample.wsdl">
            <DvakratResult>20</DvakratResult>
      </DvakratResponse>
  </soap:Body>
</soap:Envelope>
```

4.3.1 Připojení na SOAP server v prostředí .NET

Jednoduché připojení SOAP klienta v jazyce Python na SOAP serveru v prostředí .NET je možné použít knihovnu PySimpleSoap. Implementace SOAP klienta by byla totožná jako v předchozí ukázce, ale v případě volání by se použil objektový přístup k výsledkům volané webové metody.

Další možností je použití knihovny, které pracuje s SOAP serverem obsahujícím WSDL dokument. Knihovna Suds obsahuje možnost vypsání všech dostupných webových metod. V této knihovně rovněž nemusím sledovat názvy parametrů volaných webových metod, protože je knihovna převádí automaticky do správných tvarů. Jestli je název souboru totožný s názvem knihovny, tedy "suds", v takovémto případě nedojde ke správnému importování knihovny.

Client_NET.py

```
from suds.client import Client
from suds.sudsobject import Property

client = Client('http://localhost:57478/jeNazevStatu.asmx?wsdl')
```

Importuji knihovnu a vytvořím připojení pomocí WSDL dokumentu. Když za vytvořené připojení vypíšu objekt klient "print client", dostanu výpis všech dostupných webových metod a jejich očekávané parametry.

```
stat =  client.service.jestat('Krnov')
dvakrat = client.service.Dvakrat(5)
secti = client.service.Secti(a=5, b=2)
Nazdar = client.service.Nazdar()
Ahoj = client.service.Ahoj(kdo='Lucie')
```

Volání dostupných webových metod probíhá na vytvořeném objektu klient, který obsahuje v service dostupné webové metody.

```
Username = Element('Username').setText('tom')
Password = Element('Password').setText('ric')
reqsoapheader = Element('UsernameToken')
reqsoapheader.children = [Username, Password]
client.set_options(soapheaders=reqsoapheader)

print "Vynasobym dvakrat cislo 5 = ",dvakrat;
```

Posílání SOAP hlaviček probíhá pomocí přidávání elementů a metody set_options().[10] Existuje několik způsobu, jak posílání SOAP hlavičky implementovat. Tyto způsoby jsou sepsány v dokumentaci knihovny Suds. Nakonec můžu získané hodnoty vypsat pomocí proměnných, do kterých jsem je ukládal.

4.3.2 Připojení na SOAP server v PHP

Můžu použít opět knihovny PySimpleSoap nebo SUDS. V knihovně PySimpleSoap musím znova použít objektový přístup k výsledkům volané webové metody.

Další možností pro vytvoření SOAP klienta je již zmíněná knihovna Suds. Je možné vytvořit připojení na SOAP server v jazyce PHP, pomocí integrované knihovny. V tomto případě by byl SOAP klient totožný s klientem ve výše zmíněné ukázce. Při implementaci jsem nenarazil na žádné problémy, protože knihovna Suds je velmi používaná a odzkoušená v komunikaci s ostatními programovacími jazyky. Postupné vylepšení a opravy vychází právě z dotazů uživatelů této knihovny.

Připojení na SOAP server napsaném v knihovně NuSOAP, která generuje WSDL dokument je opět možné vytvořit pomocí SOAP klienta v knihovně Suds.

---

[10] Hlavička je poslaná ve tvaru :

```
<SOAP-ENV:Header>
        <UsernameToken><Username>tom</Username><Password>ric</Password></UsernameToke
</SOAP-ENV:Header>
```

Client_NuSOAP.py

```
from suds.client import Client
from suds.sudsobject import Property
client = Client('adresa SOAP serveru v NuSOAP?wsdl')

vysledekAhoj =  client.service.Ahoj('Lucie')
vysledekDvakrat =  client.service.Dvakrat(5)
vysledekSecti =  client.service.Secti(5,2)
```

4.3.3 Připojení na SOAP server v Ruby

Při připojení pomocí knihovny PySimpleSoap nebude možné zobrazit výsledek webové metody, protože SOAP odpověď neobsahuje žádný element SOAP hlavička. Kdybych tento problém chtěl obejít, musel bych přímo ve zdrojovém kódě knihovny přepsat metodu SoapDispatcher(). Další chybou v této implementaci může být zpracování parametru pomocí xsi:type="xsd:int", ale tento problém by měl být v aktuální verzi PySimpleSoap opraven.

Knihovna Suds pro jazyk Python dokáže pomocí WSDL dokumentu vytvořenou webovou služby v Soap4r knihovně provést. Funkčnost je zaručena podporou všech verzí WSDL dokumentu.

Client_soap4r.py

```
from suds.client import Client
from suds.sudsobject import Property

client = Client('http://localhost:2000/wsdl/hws.wsdl')
print client
vysledek =  client.service.hello_world(kdo = "Lucie")
print 'vysledek  :', vysledek
```

4.4 Ruby

Vytvoření webové služby pomocí SOAP protokolu v jazyce Ruby je velice problematické. Tuto možnost nedoporučuji.

Knihovna Soap4r je zastaralá a bez úprav se nedá použít od verze Ruby 1.9. Vývoj této knihovny je zastaven a dokumentace knihovny je stažena. Dokázal jsem knihovnu Soap4r zprovoznit pomocí nainstalování alternativních verzí, které obsahují opravu funkčnosti, jako například soap4r-ruby1.9, nebo Mumboe-soap4r.[11] Bohužel kvůli archaickým závislostem knihovna Soap4r, nebude fungovat v nové verzi Ruby 2.0.

Ukázku použití knihovny Saop4r, najdu například na stránce zabývající se programováním v jazyce Ruby ze stránky [20], *Tutorialspoint: Simple Easy Learning,* © 2013 http://www.tutorialspoint.com/ruby/ruby_web_services.htm.

Dalším možností je vyzkoušet knihovnu WashOut, která funguje na Rails. Knihovna WashOut umožňuje vytvořit pouze SOAP server. Rovněž zde jsem se setkal s řadou problémů. Hlavním problémem je celková nekompatibilita s některými verzemi ostatních knihoven potřebných pro správnou funkčnost knihovny WashOut. Informace o kompatibilitě jsou na adrese [24], Boris Staal, Peter Zotov. *GitHub*. 15.8.2011 https://github.com/inossidabile/wash_out.

Začnu vytvořením jednoduché SOAP webové služby pomocí knihovny Soap4r, která poběží pomocí "StandaloneServer", bez WSDL dokumentu. Nepodporuje komunikaci s ostatními jazyky, protože není správně popsaná metoda posílající SOAP požadavky. Funkce této metody jsou zastaralé.

Jelikož jsem použil upravenou knihovnu soap4r, je nutné překontrolovat cestu pro připojení knihovny.

server _soap4r.rb

```
require 'soap/rpc/standaloneServer'

class WebovaSluzba < SOAP::RPC::StandaloneServer
  def on_init
    @log.level = Logger::Severity::DEBUG
    add_method(self, 'Ahoj', 'kdo')
    add_method(self, 'Dvakrat', 'value')
  end
```

[11] Oprava chyby manuálně, přímo v souborech knihovny, postup najdete na *Rails foru*. http://railsforum.com/viewtopic.php?id=41231, tímto postupem ale neopravím všechny chyby.

```
def Ahoj(kdo)
    "Ahoj, #{ kdo }"
  end

  def Dvakrat(value)
    value*2
  end
end
```

Připojuji knihovnu Soap4r a vytvořím webovou službu s metodami Ahoj() a Dvakrat(). Logger::Severity::DEBUG umožní zobrazení veškeré komunikace na SOAP serveru, včetně SOAP požadavků a odpovědí. Add_method() je povinná metoda umožňující registraci webových metod.

```
if $0 == __FILE__
  server = WebovaSluzba.new('WS', 'urn:WS', 'localhost', 2000)
  trap(:INT) do
    server.shutdown
  end
  server.start
end
```

Provedu aktivaci vytvořené webové služby s názvem 'WS' a jmenným prostorem 'urn:WS' na localhostě s portem 2000.

client _soap4r.rb

```
require 'soap/rpc/driver'

s = SOAP::RPC::Driver.new('http://localhost:2000/', 'urn:WS')

s.add_method("Ahoj", "kdo")
s.add_method("Dvakrat", "value")
p s.Ahoj("Lucie")
p s.Dvakrat(5)
```

SOAP klient umožňuje rychlý přístup k webové službě pomocí RPC driveru. Jediný probléme je v nutnosti registrace webových metod před jejich voláním.

SOAP požadavek:

```
<?xml version="1.0" encoding="utf-8" ?>
<env:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
```

```
  <env:Body>
    <n1:Dvakrat
       env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
       xmlns:n1="urn:WS">
       <value xsi:type="xsd:int">5</value>
    </n1:Dvakrat>
  </env:Body>
</env:Envelope>
```

SOAP odpověď:

```
<?xml version="1.0" encoding="utf-8" ?>
<env:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Body>
    <n1:DvakratResponse
       env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
       xmlns:n1="urn:WS">
      <return xsi:type="xsd:int">10</return>
    </n1:DvakratResponse>
  </env:Body>
</env:Envelope>
```

4.4.1 Posílání SOAP hlaviček v SOAP4r

Posílání SOAP hlaviček probíhá pomocí vlastní třídy, která dědí z SimpleHandler. Vytvořená třída musí obsahovat metodu on_simple outbound(). Nebyl jsem schopen implementovat zpracování SOAP hlavičky na straně SOAP serveru, protože nemám k dispozici dokumentaci knihovny Soap4r.

Client_soap4r_header.rb

```
require 'soap/rpc/driver'
require 'soap/header/simplehandler'

s = SOAP::RPC::Driver.new('http://localhost:2000/', 'urn:WS')

class ClientAuthHeaderHandler < SOAP::Header::SimpleHandler
  NAMESPACE = "urn:WS"
  def initialize()
    super(XSD::QName.new(NAMESPACE, 'UsernameToken'))
    XSD::QName.new(nil, "UsernameToken")
  end

def on_simple_outbound
    return {"Username" => "tom", "Password" => "ric"}
```

```
  end
end

serv = SOAP::RPC::Driver.new(s, 'urn:WS')
serv.headerhandler << ClientAuthHeaderHandler.new
```

4.4.2 Soap4r server s WSDL dokumentem

Knihovna podporuje i používání starší verze WSDL dokumentu. Zpracování se velice podobá "Stand alone Server".[12] WSDL dokument musí byt předem připraven Soap4r knihovna neumožňuje generaci WSDL dokumentu.

server_soap4r_wsdl.rb

```
require 'soap/rpc/httpserver'

class WebovaSluzba < SOAP::RPC::HTTPServer
  def on_init
    @log.level = Logger::Severity::DEBUG
    add_method(self, 'hello_world', 'kdo')
  end

  def hello_world(kdo)
    sOut="Ahoj, #{kdo}"
  end
end

if $0 == __FILE__
  server = WebovaSluzba.new(
    :BindAddress => '0.0.0.0',
    :Port => 2000,
    :SOAPDefaultNamespace => 'http://localhost:2000/wsdl/hws.wsdl',
    :WSDLDocumentDirectory => '.'
  )
  trap(:INT) do
    server.shutdown
  end
  server.start
end
```

SOAP server vytvořený podle této ukázky přistupuje k WSDL dokumentu na adrese http://localhost:2000/wsdl/hws.wsdl.

[12] *Brendonwilson.com* 2.4.2006 http://www.brendonwilson.com/blog/2006/04/02/ruby-soap4r-wsdl-hell/.

4.4.3 Připojení na SOAP server v PHP

Používám knihovnu Savon poskytující, rozšířené možnosti manipulace SOAP zprávy. V této knihovně nemusím registrovat volanou metodu oproti knihovně Soap4r.

Savon_some.rb

```
require "savon"
client = Savon::Client.new("adresa k WSDL dokumentu")
Připojení knihovny a SOAP serveru pomocí WSDL dokumentu.
response = client.request :dvakrat do
  #soap.body = "<value>5</value>"
  soap.body = {:value => '5'}
end

p response[:dvakrat_response][:dvakrat_result]
```

První problém na který jsem narazil byl v názvu volané webové metody. Název webové metody je možné napsat přímo v uvozovkách 'Dvakrat', nebo s operátorem a malým písmenem na začátku :dvakrát. O správné přepsaní se v tomto případě stará knihovna Savon pomocí funkce snake_case.

Druhým problémem je posílání samotných parametrů. Parametr musím posílat se stejným názvem, který jsem uvedl v SOAP serveru, protože knihovna automaticky nezjišťuje ani nepřepisuje zvolené parametry.

Parametry je tedy možné napsat přímo pomocí tagů nebo pomocí operátorů. Použiji-li druhou možnost, je SOAP tělo vygenerováno automaticky za pomocí připravených pravidel v knihovně Savon.

Knihovna Savon obsahuje tři možnosti pro vypsání výsledku SOAP zprávy. Zvolím si, zda chci výsledek ve tvaru XML, HTTL, nebo Hash.

Použiju-li příkaz p response.to_hash mohu pak zjistit, jak přistoupit k výsledku přes objekty.

4.4.4 Připojení na SOAP server v prostředí .NET

Připojení SOAP klienta v jazyce Ruby pomocí knihovny Savon. Při této implementaci jsem narazil na problém týkající se jmenného prostoru, protože webová služba v prostředí .NET očekává připojení jmenného prostoru na akční element v těle SOAP. [13]

Problém lze vyřešit přidáním "xmlns" se správným jmenným prostorem.

```
require "savon"

client =
  Savon::Client.new("http://localhost:57478/jeNazevStatu.asmx?wsdl)

odpoved = client.request "jestat", "xmlns" =>
     "http://tempuri.org/" do |soap, wsdl|
  soap.body = { :s => "Krnov" }
end
odpovedDvakrat = client.request :dvakrat, "xmlns" =>
     "http://tempuri.org/" do |soap, wsdl|
  soap.body = { :value => '5' }
end

p odpovedDvakrat[:dvakrat_response][:dvakrat_result]
p odpoved[:jestat_response][:jestat_result]
```

Posílání SOAP hlavičky proběhne pomocí "Soap.header", implementované uvnitř volání webové metody.

```
odpovedGetBalance = client.request :get_balance, "xmlns" =>
"http://tempuri.org/" do |soap, wsdl|
Soap.header = { "UsernameToken" =>  {'Username' => 'tom',
                                     'Password' => 'ric'}
              }
end
```

4.4.5 Připojení na SOAP server v Pythonu

Implementace SOAP klienta v knihovně Savon, připojující se na SOAP server napsaný v jazyce Pythonu je totožná s předchozími ukázkami v této knihovně. Samozřejmě je nutné změnit adresu SOAP serveru na http://localhost:8008/.

[13] Tuto informaci jsem našel na internetové stránce [37], *Candland,* 21.6.2011 http://www.candland.net/2011/06/21/savon-rb-client-for-net-web-service/.

5 Závěr

Cílem této práce bylo zmapování možností implementace webových služeb se zaměřením na protokol SOAP, ve vybraných dynamických jazycích PHP, Pythonu, Ruby a v prostředí .NET. Druhá kapitola obsahuje teorii týkající se webových služeb a podrobné zpracování protokolu SOAP a WSDL dokumentu. Třetí kapitola popisuje knihovny použitelné v SOAP webových službách. Čtvrtá kapitola se zabývá implementaci SOAP protokolu v jednotlivých programovacích jazycích. Tato kapitola obsahuje také problémy v implementaci související s vzájemnou spoluprácí SOAP knihoven ve vybraných programovacích jazycích.

V průběhu vypracování bakalářské práce jsem postupně zdokonaloval ve vybraných programovacích jazycích a seznámil jsem se s jazykem Ruby, který jsem na začátku bakalářské práce neznal vůbec.

Podařilo se mi dosáhnout vytvoření SOAP webových služeb na všech jazycích s různou efektivitou. Implementace SOAP zpráv proběhla ve vybraných dynamických jazycích úspěšně. Spolupráce těchto jazyků proběhla rovněž úspěšně (kromě jazyka Ruby). V tomto jazyce jsem nedokázal zprovoznit vhodnou SOAP knihovnu a musel jsem používat zastaralou knihovnu Soap4r. Prověření SOAP hlaviček není úplné, protože jednotlivé knihovny ve vybraných dynamických jazycích neposkytovaly všechny funkce pro jejich používání. Posílání a zpracování SOAP hlaviček jsem dosáhl v jazyce PHP a prostředí .NET. Posílání SOAP hlaviček v Jazycích Python a Ruby je provedeno bez zpracování na SOAP serveru.

Další zkoumání v teto oblasti by mělo být zaměřeno na posílaní SOAP hlaviček a zprovoznění vhodnější knihovny v jazyce Ruby, protože je posílání SOAP hlaviček v některých případech neúplné.

Použitá literatura

[1] Webová služba. z: *Wikipedie: otevřená encyklopedie* [online]. Wikimedia Foundation, 10.3.2013 [cit. 2013-05-02]. Dostupné z: http://cs.wikipedia.org/wiki/Webová_služba

[2] Web Services Description Language. In: *Wikipedie: the free encyclopedia* [online]. Wikimedia Foundation, 2.5.2013 [cit. 2013-05-02]. Dostupné z: http://en.wikipedia.org/wiki/WSDL

[3] WSDL Tutorial. *W3schools.com* [online]. © 1999-2013 [cit. 2013-05-02]. Dostupné z: http://www.w3schools.com/wsdl/

[4] Web Services Description Language (WSDL) 1.1. *W3C* [online]. 15.3.2001 [cit. 2013-05-02]. Dostupné z: http://www.w3.org/TR/wsdl

[5] SOAP Version 1.2 Part 1: Messaging Framework. *W3C* [online]. 27.4.2007 [cit. 2013-05-02]. Dostupné z: http://www.w3.org/TR/2007/REC-soap12-part1-20070427/

[6] SOAP Tutorial. *W3schools.com* [online]. © 1999-2013 [cit. 2013-05-02]. Dostupné z: http://www.w3schools.com/soap/

[7] SOAP. In: *Wikipedie: the free encyclopedia* [online]. Wikimedia Foundation, 18.4.2013 [cit. 2013-05-02]. Dostupné z: http://en.wikipedia.org/wiki/Simple_Object_Access_Protocol

[8] NuSOAP: SOAP Toolkit for PHP. *Nusoap.sourceforge.net* [online]. © 1999-2009 [cit. 2013-05-02]. Dostupné z: http://nusoap.sourceforge.net/

[9] SOAP. *PHP* [online]. © 2001-2013 [cit. 2013-05-02]. Dostupné z: http://us.php.net/manual/en/book.soap.php

[10] Web Services. *Python* [online]. 29.9.2012 [cit. 2013-05-02]. Dostupné z: http://wiki.python.org/moin/WebServices

[11] PySimpleSOAP. *Pysiimplesoap: Python Simple SOAP Library* [online]. [cit. 2013-05-02]. Dostupné z: http://code.google.com/p/pysimplesoap/

[12] Zolera SOAP Infrastructure (ZSI). *Python Web Services* [online]. [cit. 2013-05-02]. Dostupné z: http://pywebsvcs.sourceforge.net/

[13] SUDS. *Suds* [online]. © 2003-2012, 1.1.2012 [cit. 2013-05-02]. Dostupné z: https://fedorahosted.org/suds/

[14] What is Spyne. *Spyne* [online]. [cit. 2013-05-02]. Dostupné z: http://spyne.io/

[15] Soaplib. *GitHub: Soaplib v 2.0 beta documentation* [online]. © 2010 [cit. 2013-05-02]. Dostupné z: http://soaplib.github.io/soaplib/2_0/

[16] What is Ladon. *Ladon Webservice* [online]. [cit. 2013-05-02]. Dostupné z: http://ladonize.org/index.php/Main_About

[17] A SOAP/XML Schema Library for Python. ELMAN, Adam. *SOAPy* [online]. 26.4.2001 [cit. 2013-05-02]. Dostupné z: http://soapy.sourceforge.net/

[18] SOAP Clients. *The Ruby toolbox* [online]. [cit. 2013-05-02]. Dostupné z: https://www.ruby-toolbox.com/categories/soap

[19] SOAP. *RubyGems.org: your community gem host* [online]. 1.7.2009 [cit. 2013-05-02]. Dostupné z: https://rubygems.org/search?utf8=%E2%9C%93&query=SOAP

[20] Web Services with Ruby: SOAP4R. *Tutorialspoint: Simple Easy Learning* [online]. © 2013 [cit. 2013-05-02]. Dostupné z: http://www.tutorialspoint.com/ruby/ruby_web_services.htm

[21] Ruby + SOAP4R + WSDL Hell. *Brendonwilson.com* [online]. 2.4.2006 [cit. 2013-05-02]. Dostupné z: http://www.brendonwilson.com/blog/2006/04/02/ruby-soap4r-wsdl-hell/

[22] Savon: Version 1. *Savon: Heavy metal SOAP client* [online]. 6.10.2009 [cit. 2013-05-02]. Dostupné z: http://savonrb.com/version1.html

[23] Handsoap. Unwire A/S. *GitHub* [online]. 1.4.2009 [cit. 2013-05-02]. Dostupné z: https://github.com/unwire/handsoap

[24] WashOut. In: STAAL, Boris a Peter ZOTOV. *GitHub* [online]. 15.8.2011 [cit. 2013-05-03]. Dostupné z: https://github.com/inossidabile/wash_out

[25] Web Services Description Language Tool: Wsdl.exe. *Msdn* [online]. © 2013 [cit. 2013-05-02]. Dostupné z: http://msdn.microsoft.com/en-us/library/vstudio/7h3ystb6(v=vs.100).aspx

[26] ServiceModel Metadata Utility Tool: Svcutil.exe. *Msdn* [online]. 8.2.2012 [cit. 2013-05-02]. Dostupné z: http://msdn.microsoft.com/en-us/library/aa347733.aspx

[27] How to Consume a Web Service. BEEN, John. *This Is Not a Blog* [online]. 17.5.2009 [cit. 2013-05-02]. Dostupné z: http://johnwsaunders3.wordpress.com/2009/05/17/how-to-consume-a-web-service/

[28] PHP SOAP Extension. STOGOV, Dmitry. *Zend Developer Zone: Advancing the art of PHP* [online]. 30.11.2001 [cit. 2013-05-02]. Dostupné z: http://devzone.zend.com/25/php-soap-extension/

[29] Authenticate .NET Web Service with Custom SOAP Header. SHOKR, Ahmed. *Code Project: For those who code* [online]. 29.6.2008 [cit. 2013-05-02]. Dostupné z: http://www.codeproject.com/Articles/27365/Authenticate-NET-Web-Service-with-Custom-SOAP-Head

[30] Calling PHP SOAP Server from C#/.NET Client. *Nilzor's Techblog* [online]. 5.6.2010 [cit. 2013-05-02]. Dostupné z: http://www.nilzorblog.com/2010_05_01_archive.html

[31] PHP: Soap Client calling .NET Web service. *Coding Friends* [online]. 16.4.2010 [cit. 2013-05-02]. Dostupné z: http://www.codingfriends.com/index.php/2010/04/16/soap-client-calling-net-web-service/

[32] Process SOAP Headers in PHP. *Stackoverflow* [online]. 22.12.2009 [cit. 2013-05-02]. Dostupné z: http://stackoverflow.com/questions/1948897/process-soap-headers-in-php

[33] PHP Using Authentication with SOAP. *Handbook* [online]. [cit. 2013-05-02]. Dostupné z: http://www.bitpapers.com/2012/04/php-using-authentication-with-soap_09.html

[34] PHP Generating a SOAP Header. *Handbook* [online]. [cit. 2013-05-02]. Dostupné z: http://www.bitpapers.com/2012/04/php-generating-soap-header.html

[35] PHP Webservices and C# / .NET SOAP Clients. *Sanity Free Coding: Methods to the Madness* [online]. 3.7.2006 [cit. 2013-05-02]. Dostupné z: http://www.sanity-free.org/125/php_webservices_and_csharp_dotnet_soap_clients.html

[36] Sample Code. *Adobe: Developer Connection* [online]. © 1996-2011 [cit. 2013-05-02]. Dostupné z: https://developer.omniture.com/en_US/documentation/omniture-administration/c-api-admin-sample

[37] Savon.rb client for .Net web service.: .net, rails, ruby, savon, soap. GITHUB. *Candland* [online]. 21.6.2011 [cit. 2013-05-02]. Dostupné z: http://www.candland.net/2011/06/21/savon-rb-client-for-net-web-service/

Seznam příloh

Seznam obrázků

Instalace PHP

1. Nainstalujte wampserver2.2iEN.

   Instalaci najdete na adrese

   http://www.stahuj.centrum.cz/internet_a_site/servery/databazove/wampserver/
2. V souboru php.ini odkomentujte řádek s extension=php_soap.dll. Soubor je umístěn v adresáři wamp\bin\apathe

Instalace Pythonu

1. Nainstalujte python-2.7.3.msi.

   Instalaci najdete na adrese www.python.org/ftp/python/2.7.3/python-2.7.3.msi
2. Nainstalujte python-setuptools, pomocí ez_setup.py, který naleznete v adresáři instalace.
3. Nainstalujte PySimpleSOAP-1.05a.win32.exe.

   Instalaci najdete na adrese http://code.google.com/p/pysimplesoap/downloads/list

   Nebo můžete knihovnu nainstalovat manuálně. Pomocí příkazového řádku najedete do složky instalace\pysimplesoap-1.05, kde najdete zdrojový kód knihovny. Do příkazového řádku napište "python setup.py install".
4. Upravte soubor client.py, změnte TIMEOUT z 60 na None.

   Soubor client.py najdete v adresáři Pythonu.

   Lib\site-packages\pysimplesoap
5. Nainstalujte knihovnu Suds

   V adresáři instalace\python-suds-0.3.7 najdete setup.py, který opět spustíte přes příkazový řádek pomocí příkazu "setup.py install".

Instalace Ruby

1. Nainstaluj ruby rubyinstaller-1.9.3-p327.exe

   Instalaci najdete na adrese http://rubyinstaller.org/downloads/
2. Nainstalujte rubygems-1.8.24, který najdete v adresáři instalace

   Do příkazového řádku v tomto adresáři napište příkaz ruby "setup.rb install".

   (možná budete potřebovat příkazový řádek zapnout jako administrátor)
3. Následně můžete instalovat knihovny příkazy:

   gem install soap4r , gem install mumboe-soap4r, gem install soap4r-ruby1.9

   gem install savon

   gem instar wash_out, gem install rails

Instalace Microsoft Visual studio

1. Nainstalovaný Microsoft Visual studio 2010.

   Instalaci najdete na adrese http://www.microsoft.com/visualstudio/cze/downloads

Instalovaní knihoven, můžete provést pomocí dávkového souboru instalace.bat.

Spuštění webových služeb

Instrukce pro spuštění webových služeb a jejich testování jsou rozsáhlé, proto byl vytvořen pomocný dávkový soubor sluzby.bat. Spuštění webových služeb provádějte podle instrukcí v tomto dávkovém souboru.

Podrobné pokyny pro spuštění webových služeb najdete v adresáři Dokumentace s názvem "Spusteni SOAP.pdf"